8Ч

УДК 004.4'42

***Ю.В. Чернухин, М.Ю. Поленов, Д.В. Булгаков***
Технологический институт Южного федерального университета, г. Таганрог, Россия
chernukhin@dce.tsure.ru, polenov@dce.tsure.ru, buda@mail.ru

# Использование мультиязыковой трансляции при конверсии моделей, представленных на языках описания аппаратуры

Рассматривается подход к применению разработанной ранее среды многоязыковой трансляции моделей (Мультитранслятора) для перевода программных проектов, представленных на языках описания аппаратуры различных уровней. Развитие данного подхода позволяет использовать Мультитранслятор в качестве средства кросс-трансляции проектов для моделирования в системотехнических САПР.

## Введение

Интенсивное развитие микроэлектроники привело в последние годы к появлению и широкому использованию новой технологии проектирования электронных устройств на базе микропроцессоров – разработке систем на кристалле (System-on-Chip, SoC), которые содержат программируемые процессорные ядра, специализированные логические блоки, модули памяти, интерфейсные и периферийные устройства, аналоговые и аналого-цифровые схемы.

Создание систем на кристалле (СНК) является достаточно универсальной и многоуровневой технологией, объединяющей в себе методы проектирования аппаратно-программных комплексов и встраиваемых систем на основе стандартных процессоров и процессорных ядер, разработки встроенного программного обеспечения, программируемых логических интегральных схем (ПЛИС), полузаказных и заказных интегральных схем [1].

Степень интеграции современных СНК достигает нескольких десятков миллионов вентилей на кристалле, и для их разработки и моделирования используются новые подходы, методы и средства проектирования системотехнического уровня.

Основные этапы разработки систем на кристалле традиционно реализуются различными средствами проектирования при помощи разных языков описания проектов [2]. Так, архитектурно-алгоритмическое проектирование и аппаратное моделирование реализуются на языках С/С++ [3] и SystemC [4]; поведенческое моделирование, функциональная верификация и тестирование – на SystemC и языках описания аппаратуры (Hardware Description Language, HDL), обычно VHDL [5]; логическое моделирование и схемотехническое проектирование – на языках описания аппаратуры VHDL и Verilog.

Таким образом, для обеспечения эффективности процесса разработки СНК актуальным является решение задачи кросс-трансляции проектов для языков, используемых на различных этапах проектирования. Традиционное решение данной задачи требует разработки набора полных трансляторов для каждой пары языков, что является весьма трудоемким и длительным процессом.

В связи с этим с целью сокращения времени создания трансляторов и обеспечения возможности их модификации для новых версий транслируемых языков аппаратуры предлагается использовать методы и разработанную ранее среду многоязыковой трансляции – Мультитранслятор [6]. Мультитранслятор уже показал свою эффективность для решения задач моделирования [7] и для работы в качестве компилятора компиляторов [8]. По этой причине постановка задачи исследования возможностей использования средств многоязыковой трансляции и в области САПР вычислительных устройств весьма целесообразна.

# 1. Языковые средства разработки систем на кристалле

Для реализации полного цикла проектирования систем на кристалле используется определенный набор программных продуктов и языков программирования [2]. Рассмотрим их подробнее.

Проектирование системы начинается с проработки ее архитектуры на языках высокого уровня, где традиционно используются языки C/C++. На этом этапе производится разбиение системы на системные блоки, выполняется проработка сложных алгоритмов и их отладка, представляется разделение на аппаратные и системные блоки и интерфейсы между ними. В итоге получается высокоуровневая поведенческая модель функционирования устройства, решающая поставленные разработчиком задачи. После определения аппаратных блоков производится их функциональное проектирование и моделирование на RTL-уровне (Register Transfer Level – уровне регистровых пересылок). Здесь применяются языки описания аппаратуры (ЯОА) VHDL и Verilog. Далее производится логический и физический синтез проектируемого устройства с получением прототипа СНК.

Языки программирования C/C++ обычно используются в качестве основного средства разработки алгоритмического представления и создания моделей высокого уровня абстракции – поведенческих моделей СНК. Основные достоинства C/C++ – это широкая распространенность и сравнительно низкая стоимость, доступность средств программирования, простота в освоении и использовании, а главный недостаток – отсутствие специализированных библиотек системного уровня, вследствие чего поведенческие модели аппаратуры приходится создавать практически вручную [9].

Язык SystemC представляет собой расширение стандартного языка программирования C++, реализованное в виде отдельных библиотек специальных классов. Данные библиотеки содержат в себе конструкции, позволяющие создавать эффективные и точные модели программных алгоритмов, аппаратных архитектур, интерфейсов и схем на системном уровне, т.е. практически всех компонентов встроенных систем [9]. Использование SystemC значительно упрощает процесс перехода от архитектурной поведенческой модели на C++ к RTL-модели на ЯОА, например, VHDL или Verilog. Но, как правило, разработчики, хорошо знакомые с языками описания аппаратуры типа VHDL/Verilog, не очень хорошо разбираются с программированием на языках высокого уровня, подобным C++. Таким образом, использование SystemC, совмещающего возможности языков высокого уровня и имеющего специальные конструкции для описания аппаратуры, является способом ускорения и оптимизации процесса проектирования СНК.

Язык SystemC, однако, не лишен недостатков, поскольку описание схемы на языке C не всегда удобно для проектировщиков, работающих с ЯОА, а существующие средства синтеза с SystemC являются в основном коммерческими продуктами. Следовательно, разработчики аппаратуры нуждаются в конверсии моделей с SystemC в VHDL/Verilog модели [10].

Таким образом, возникает необходимость в разработке программных модулей для трансляции (конверсии) проектов, написанных на SystemC в проекты на VHDL. Создание таких модулей позволит разработчикам использовать все возможности отладки поведенческих моделей на C++, переходя в дальнейшем к RTL-модели на SystemC, и при этом иметь возможность получить после трансляции готовый прототип проекта системы на кристалле для перехода к средам проектирования, использующим в качестве основного языка описания язык VHDL.

## 2. Трансляция моделей систем на кристалле с языка SystemC в VHDL

При использовании для трансляции моделей систем на кристалле традиционного подхода к построению трансляторов разработчики могут столкнуться со следующими трудностями:

– в состав транслятора должно входить большое количество модулей, выполняющих ту или иную подзадачу трансляции, написание которых является трудоемким, требует значительных временных затрат и многочисленную команду разработчиков;

– транслятор строится для единственного языка моделирования и оптимизируется под этот входной и требуемый выходной языки;

– транслятор обычно работает только с программами для определенной аппаратной платформы или заданной операционной системы и не содержит средств коррекции процесса трансляции для иных платформ, даже если различия в их исходном коде не носят принципиального характера;

– при внесении поправок в схему трансляции языков при изменении версии языка необходима коррекция всех (или большинства) модулей транслятора, что ведет к появлению большого количества ошибок и требует усилий по координации действий разработчиков.

Предлагаемый и реализованный авторами подход к решению данных проблем основан на применении среды многоязыковой трансляции – Мультитранслятора (МТ) для создания трансляторов с языков описания аппаратуры [11]. МТ для этих целей позволяет значительно сократить количество промежуточных этапов разработки транслятора, уменьшить время разработки и сократить время на создание трансляторов для новых версий транслируемых языков [6].

Традиционная организация транслятора предполагает наличие нескольких обязательных этапов, причем на каждом из этих этапов происходит преобразование исходной программы из одного промежуточного представления в другое. Типичными этапами трансляции являются: лексический анализ, грамматический анализ, семантический анализ и генерация кода [12]. Для мультитрансляции характерен отличный подход, при котором этапы лексического анализа и грамматического разбора выполняются универсальным ядром Мультитранслятора, а полную информацию о грамматике входного языка и соответствующих конструкциях выходного языка несет подключаемый к ядру трансляционный модуль (ТМ), реализующий отчасти функции генератора кода.

Мультитранслятор позволяет создавать трансляционные модули, которые описывают правила перевода исходных кодов проектов систем с языков описания аппаратуры в целевые коды. ТМ Мультитранслятора формируется на языке описания грамматик в виде множества грамматических правил исходного языка. Также в этих правилах посредством специального языка описания действий задаются необходимые действия по обработке исходного кода, выполняемые ядром Мультитранслятора, и

генерации выходного кода. Встроенный компилятор Мультитранслятора позволяет сгенерировать на основании подключенного трансляционного модуля выходной исполняемый файл частного транслятора для выбранной пары языков, который может использоваться как самостоятельный модуль трансляции.

Важной особенностью МТ является то, что при появлении новой версии языка описания аппаратуры не требуется создание нового транслятора «с нуля». Достаточно модифицировать исходный код уже существующего трансляционного модуля и откомпилировать его с помощью ядра МТ. Кроме того, МТ позволяет использовать не все множество грамматических правил исходного языка описания аппаратуры, а необходимое подмножество как для решения частных задач трансляции языков описания аппаратуры, так и для исследования возможности создания полноценных трансляторов для выбранной пары языков в дальнейшем. Например, для языка SystemC возможно описание подмножества конструкций, которые широко используются при проектировании вычислительных устройств, а не всего набора операторов.

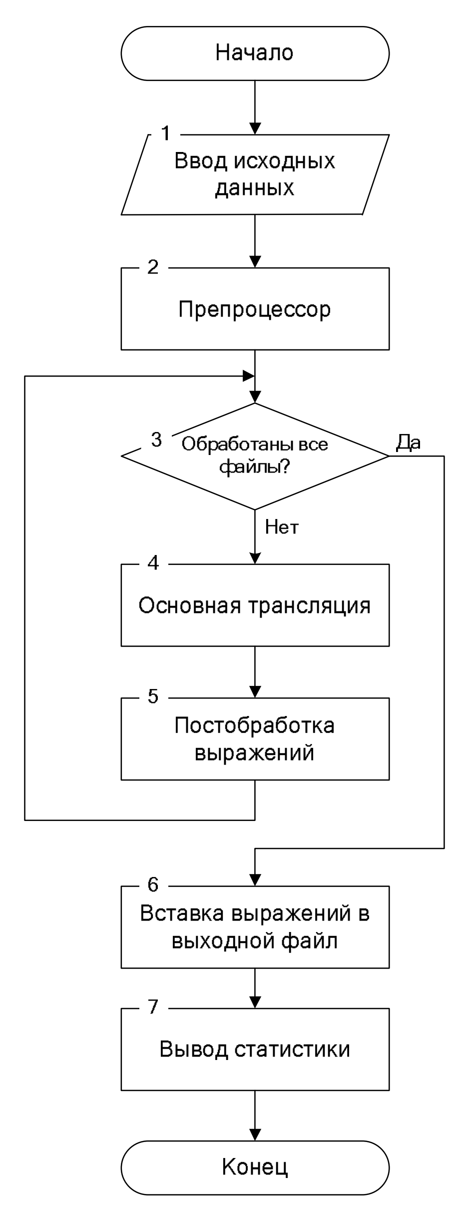



Рисунок 1 – Основные этапы трансляции проектов на ЯОА

При решении задачи по исследованию возможности построения трансляционных модулей для языков описания аппаратуры в среде Мультитранслятор была выбрана пара языков SystemC и VHDL. Как было рассмотрено ранее, выбор этой пары языков был обусловлен необходимостью преобразования кодов на языке SystemC, как языке, дающем максимум возможностей по созданию поведенческой модели устройства и с возможностью описания на уровне RTL, в коды на языке VHDL, обладающем более широкими возможностями логического и физического синтеза систем на кристалле.

Рассмотрим особенности языков SystemC и VHDL и условия возможности их взаимной трансляции.

Язык SystemC является расширением языка C++, и, следовательно, он целиком поддерживает парадигму объектно-ориентированного программирования с присущими ей объектами и классами. Поскольку объектно-ориентированная версия языка VHDL только развивается и еще малоизвестна широкому кругу разработчиков, то за основу синтаксиса VHDL был взят стандарт IEEE 1076, который не предусматривает использование классов [5].

Весь процесс трансляции для языков описания аппаратуры можно представить в виде алгоритма, изображенного на рис. 1. Как видно из рис. 1, до начала основного процесса трансляции производится препроцессорная обработка, цель которой – найти подключенные файлы проекта. Далее последовательно транслируются все найденные файлы. Причем трансляция файла производится в три этапа:

1. Основная трансляция.
2. Трансляция выражений – проверка выражений на вхождение во множество допустимых операций и доопределение функциями приведения типов.
3. Вставка готовых выражений вместо меток в выходной файл.

При трансляции моделей с SystemC как для препроцессора, так и для основной трансляции и трансляции выражений используются отдельные наборы грамматических правил языка SystemC, которые формируются на основе синтаксиса языков C++ и SystemC.

# 3. Разработка трансляционного модуля Мультитранслятора с языка SystemC в VHDL

В рамках рассматриваемого подхода к использованию Мультитранслятора для трансляции проектов моделей на языки описания аппаратуры, в среде МТ был разработан трансляционный модуль для перевода проектов с языка SystemC на язык VHDL.

Данный модуль был реализован в виде набора продукционных правил, написанных на языках описания грамматик и действий МТ на основе грамматики языка SystemC [4], [9] и сформированных действий по генерации выходного кода на VHDL.

Рассмотрим фрагмент трансляционного модуля, описывающий обработку условного оператора *if*:

```
rule <"if_statement">
{
     before:
          strIfLine="";
     variant
     {
          symbol "if" {strIfLine = strIfLine+"if ";}
          symbol "("  {}
          symbol <"condition">
```

```
        {
            //Извлечение из стека строки условия
            string strExpr = PopStr();
            strIfLine = strIfLine+ strExpr +" then";
        }
        symbol ")"
        {
            //Вывод в файл перед меткой оператора if
        InsertBeforeTextInVHDFile(strIfLine,strDefBodyLabel,Level,1);
        }
        symbol [<"else_statement">] {}

        InsertBeforeTextInVHDFile("end if;",strDefBodyLabel,Level,1);
    }
}
```

Как видно из приведенного правила, передача значений между уровнями дерева вывода осуществляется посредством стека. Так внутри нетерминального символа *<"condition">* формируется строка условия, которая помещается в стек. При выходе из этого символа производится извлечение из стека выражения, которое в дальнейшем используется в формировании новой строки выходного файла.

Важным моментом также является определение ограничений для входных конструкций. Это связано с тем, что не все конструкции, поддерживаемые SystemC, возможно транслировать на язык VHDL. Так, например, в VHDL не поддерживаются глобальные переменные. Их поддержка реализована только стандартом IEEE 1076-1993 [5]. Но при трансляции возможна их замена на сигналы с соответствующим именем, которые являются глобальными для всех процессов модуля. В этом случае в процессах, использующих глобальные переменные на VHDL, следует объявить локальные переменные, запись в которые начального значения из соответствующего глобального сигнала производить перед входом в тело процесса. По выходу из тела процесса следует возвратить значение локальной переменной глобальному сигналу.

Необходимо также определиться с использованием библиотек в модулях VHDL. Так как каждая подключенная библиотека содержит набор объектов и операции над ними, то количество и номенклатура библиотек должны быть подобраны с целью покрытия наибольшего количества арифметических и логических операций, типов объектов языка SystemC. Остальные операторы и объекты SystemC, которые не поддерживаются библиотеками VHDL, должны быть либо заменены аналогичными с соответствующими ограничениями на использование, либо помечены как недопустимые для трансляции.

В общем, набор транслируемых конструкций модуля можно разделить на ряд подклассов:
- структура модулей;
- выражения;
- операторы;
- структурное описание (соединения компонентов).

В отличие от SystemC, язык VHDL является строго структурированным в плане расположения интерфейсной части устройства (раздел *entity*) и разделов описания моделей. В моделях на SystemC порты могут располагаться в любом порядке относительно объявления других компонентов класса-модуля (сигналов, функций, процессов). Эта особенность влияет на способ организации перевода моделей, поскольку в качестве процедуры грамматического разбора используется поиск в глубину с возвратами (Backtrack) [6], [13]. Таким образом, формирование выходного кода является последователь-

ным процессом и производится слева направо и при начале трансляции проекта на SystemC формируется общая структура модуля на VHDL с вспомогательными текстовыми метками, обозначающими разделы выходного модуля.

В дальнейшем по мере распознавания конструкций входного языка на места соответствующих меток кода выходного языка помещаются оттранслированные конструкции. На рис. 2 изображена условная схема одного из вариантов разбора конструкций (слева) языка SystemC и расположение соответствующих выходных конструкций в выходном тексте на VHDL. Как видно из рис. 2, при распознавании заголовка модуля происходит вывод разделов модуля выходного кода с соответствующими метками. В случае распознавания объявления процесса, в выходной файл в раздел архитектурного тела помещаются ключевые слова, описывающие данный процесс, а вместо тела процесса размещается соответствующая метка для последующей вставки конструкций реализации процесса.

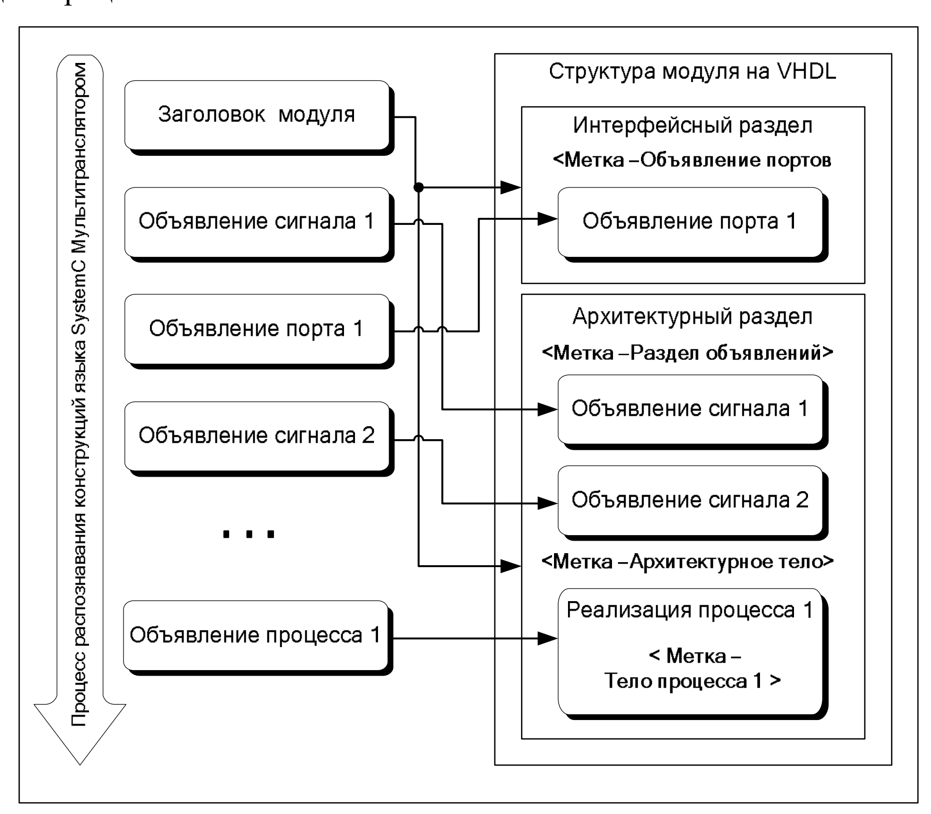



Рисунок 2 – Условная схема разбора конструкций

Как видно из рис. 2, вставка меток осуществляется с использованием имен конкретных элементов конструкций, например, процессов, и в данном случае следует организовать специальные таблицы хранения информации об объектах (имена и принадлежность к конкретным модулям). Для организации таких таблиц в МТ возможно использование словарей, связывающих определенное значение, называемое ключом, с другим значением, называемым результатом ключа [13]. Возможно также использование внешних элементов для хранения данных, например, возможно подключение сторонних библиотек с функциями, написанными на языках высокого уровня. Так, например, для хранения таблиц портов каждого трансляционного модуля могут быть организованы динамические списки иерархической структуры, организованные с помощью стандартного класса на Delphi *TList*.

Подключение внешней функции *MainLibrary.dll* к трансляционному модулю МТ может быть осуществлено следующим образом:

```
extern string GetClassName(int nCLIdx) library "Library\\MainLibrary.dll",
"GetClassName".
```

Рассмотрим также особенности трансляции выражений. Язык SystemC, в отличие от VHDL, позволяет выполнять операции над именами различных типов с использованием неявного приведения типов. Для операндов в VHDL набор операций ограничен. Типы операндов и допустимые операции заданы в подключаемых библиотеках. Рассмотрим пример оператора присваивания:

```
A = B + C + D;,
```

где операнды A, D – знаковые целые (IR); B, C – векторы бит (SV). Тогда данный оператор можно представить в виде присваивания выражения, состоящего из типов операндов:

```
IR = SV + SV + IR.
```

Допустим, что в подключенных библиотеках выходного языка определены следующие операции:

```
SV = SV + SV;
SV = IR + SV;
SV = SV + IR;
IR = IR + IR;
```

Тогда рассматриваемый оператор присваивания можно представить в виде дерева разбора, которое изображено на рис. 3. Из рис. 3 видно, что операция сложения двух операндов типа бит-вектор (SV) может давать в сумме объект типа бит-вектор (SV). Далее должна быть произведена операция суммы бит-вектора и целого знакового (IR), чтобы в результате было получено целое знаковое. Но в наборе операций выходного языка нет суммирования *IR=SV+IR*. Отсюда следует, что операнд типа бит-вектор следует доопределить функцией приведения в целое знаковое, которое в сумме с целым знаковым даст результат требуемого типа.

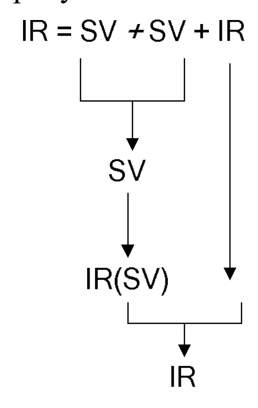



Рисунок 3 – Дерево разбора оператора присваивания

Аналогичным образом возможен разбор любого выражения входного языка, написанного на SystemC, с получением выходного выражения на VHDL при помощи функций приведения типов. Для доопределения выражений был введен дополнительный этап трансляции, на котором выполнялась трансляция выражений, сформированных в промежуточном файле на этапе основной трансляции. После этого готовые выраже-

ния вставлялись в выходной файл. Трансляция управляющих конструкций входного языка осуществлялась простой заменой ключевых слов SystemC на ключевые слова VHDL с использованием таблицы соответствий операторов.

Поскольку в языках описания аппаратуры помимо поведенческого описания, определяющего логику работы блоков, к числу основных видов относится структурное описание, то необходимо рассмотреть специфику его трансляции. Для структурного описания характерна связка портов отдельных устройств (компонентов) посредством сигналов архитектурного тела. При этом возможны различные варианты появления компонента в модуле и соответствующие действия транслятора:

– если компонент описан в одном из транслируемых модулей проекта, производится простое связывание портов, название и типы которых фиксируются в процессе трансляции подключаемых компонентов;
– если компонент является системным, то транслятор может взять имена и типы портов аналогичного компонента из составленных заранее таблиц соответствия системных компонент транслируемых языков;
– если компонент не присутствует в данных таблицах, то решение проблемы связывания портов может быть переложено на плечи разработчика. При этом в процессе трансляции при распознании неизвестного компонента выводится диалоговое окно, которое может быть реализовано с помощью языка высокого уровня, например Delphi, и подключенное в виде внешней библиотеки. В этом диалоговом окне пользователь может указать известный ему VHDL-аналог компонента на SystemC с соответствующими портами и связать порты с сигналами модуля.

Разработанный трансляционный модуль трансляции проектов моделей с языка SystemC на VHDL был протестирован на группе устройств, таких как регистр, счетчик, АЛУ, процессорный элемент. В табл. 1 приведены фрагменты исходного кода процесса для синхронного регистра на SystemC и соответствующего сгенерированного процесса на VHDL.

Таблица 1

<table>
<tr><th>Исходный фрагмент кода на SystemC</th><th>Сгенерированный МТ фрагмент кода на VHDL</th></tr>
<tr><td><pre>void dffa::do_ffa()
{
     if (reset)
     {
          dout = false;
     } else if (clock.event())
          {
          dout = din;
          }
};</pre></td><td><pre>--Function do_ffa()
 do_ffa: process(clock,reset)
     begin
     if (clock'event and clock=true)
         OR reset'event then
          --Process body
          if reset then
               dout<=false;
          else
          if clock'event then
               dout<=din;
          end if;
          end if;
     end if;
  end process;</pre></td></tr>
</table>

Работоспособность сгенерированных Мультитранслятором моделей была проверена в среде Active-HDL [14], получены временные диаграммы, подтверждающие корректную работу трансляционного модуля с языка SystemC на VHDL.

# Заключение

На основании полученных результатов можно сделать вывод о корректной работе трансляционного модуля с языка SystemC на язык VHDL и, следовательно, о возможности создания таких модулей для трансляции других языков описания аппаратуры. Кроме трансляционного модуля прямого перевода моделей с SystemC на VHDL, также была исследована возможность создания трансляционного модуля обратного преобразования проектов с VHDL на SystemC и получены результаты, говорящие о том, что применение Мультитранслятора и его трансляционных модулей достаточно эффективно для решения задач проектирования систем на кристалле.

# Литература

1. Бухтеев А.В. Методы и средства проектирования систем на кристалле / А.В. Бухтеев // Chip news. – 2003. – № 4. – С. 4-14.
2. Шагурин И.В. Применение языка SystemC и средств разработки на его основе для проектирования «Систем на кристалле» / И.В. Шагурин, В.А. Каньшев // Инженерная практика. – 2006. – Т. 32, № 9. – С. 23-32.
3. Марченко А.Л. С++. Бархатный путь / Марченко А.Л. – М. : Горячая линия-Телеком, 2005. – 399 с.
4. SystemC. Version 2.0 User's Guide [Электронный ресурс]. – Режим доступа : http://www.systemc.org
5. Welcome to the VHDL. Language Guide [Электронный ресурс]. – Режим доступа : http://ece.wpi.edu/~wrm/Courses/EE3810/geninfo/Welcome%20to%20the%20VHDL%20Language.pdf
6. Чернухин Ю.В. Интерактивная среда мультиязыковой трансляции сложных программных моделей / Ю.В. Чернухин, М.Ю. Поленов, Р.В. Фадеев // Анализ и моделирование развивающихся интеллектуальных систем : межвуз. сб. науч. трудов. – Ростов н/Д : Изд-во СКНЦ ВШ, 2003. – Вып. 4. – С. 10-20.
7. Чернухин Ю.В. Инструментальные средства импорта моделей виртуальных моделирующих сред / Ю.В. Чернухин, В.Ф. Гузик, М.Ю. Поленов // Искусственный интеллект. – 2006. – № 4. – С. 59-65.
8. Чернухин Ю.В. Возможности применения среды мультитрансляции в качестве компилятора компиляторов / Ю.В. Чернухин, М.Ю. Поленов, Д.В. Левченко // Искусственный интеллект. – 2008. – № 4. – С. 712-720.
9. Знакомство с SystemC [Электронный ресурс]. – Режим доступа : http://systemc.dax.ru/book/1.html
10. Смешков А.С. Автореферат выпускной работы магистра на тему: «Интерактивный вспомогательный модуль (TLM Wizard) для генерации начального кода цифровых устройств, базирующихся на технологии transaction level modeling (TLM)» [Электронный ресурс] / А.С. Смешков. – Режим доступа : http://masters.donntu.edu.ua/2007/fvti/smeshkov/diss/index.htm
11. Чернухин Ю.В. Многоязыковая трансляция для моделирующих систем и САПР / Ю.В. Чернухин, М.Ю. Поленов // Труды Международных научно-технических конференций «Интеллектуальные системы» (AIS'07) и «Интеллектуальные САПР» (CAD-2007). – М. : Физматлит, 2007. – Т. 1. – С. 228-233.
12. Ахо А. Теория синтаксического анализа, перевода и компиляции / А. Ахо, Дж. Ульман. – М. : Мир, 1979. – Т. 1. – 612 с.
13. Чернухин Ю.В. Исследование продукционных систем искусственного интеллекта на программном комплексе «Мультитранслятор» : [учебное пособие] / Чернухин Ю.В., Гузик В.Ф., Фадеев Р.В. ; под ред. Ю.В. Чернухина. – Таганрог : Изд-во ТРТУ, 2005. – 145 с.
14. Active-HDL. Aldec, Inc. [Электронный ресурс]. – Режим доступа : http://www.aldec.com/ActiveHDL

***Ю.В. Чернухін, М.Ю. Полєнов, Д.В. Булгаков***

**Використання мультимовної трансляції при конверсії моделей, представлених мовами опису апаратури**

Розглядається підхід до використання розробленого раніше середовища багатомовної трансляції моделей (Мультитранслятора) для перекладу програмних проектів, представлених мовами опису апаратури різних рівнів. Розвиток даного підходу дозволяє використовувати Мультитранслятор як засіб крос-трансляції проектів для моделювання у системотехнічних САПР.

***Yu.V. Chernukhin, M.Yu. Polenov, D.V. Bulgakov***

**Multilanguage Translation Usage at Conversion of Models Presented on Hardware Description Languages**

The approach to application of previously developed the multilanguage models translation environment (Multitranslator) for conversion of program projects presented on different levels hardware description languages is considered. Development of the given approach allows to use the Multitranslator as projects cross translator for simulation and EDA systems.

*Статья поступила в редакцию 26.06.2009.*